Protocol Audit Report

Version 1.0

Cyfrin.io

December 9, 2023

Protocol Audit Report

Kwame 4B

5th December, 2023

Prepared by: [Kwame] Lead Security Researcher: - Kwame 4b

Table of Contents

Protocol Summary

PasswordStore is a protocol that is designed for a single user to be able to store and access his password later, Incase he forgets.

Disclaimer

Kwame4B makes all effort to find as many vulnerabilities in the code in the given time period, but holds no responsibilities for the findings provided in this document. A security audit by the team is not an endorsement of the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the Solidity implementation of the contracts.

Risk Classification

		Impact		
		High	Medium	Low
	High	H	H/M	M
Likelihood	Medium	H/M	M	M/L
	Low	M	M/L	L

We use the CodeHawks severity matrix to determine severity. See the documentation for more details.

Audit Details

The findings described in this document correspond the following commit hash

```
7d55682ddc4301a7b13ae9413095feffd9924566
```

Scope

```
./src/PasswordStore.sol
```

Roles

- Owner: The user who can store and read the password later
- Outsiders: No one else should be able to set or read the password

Executive Summary

- Audit was a simple codebase i understood was able to get some highs and some gas issues
- we spent 3.5 hours reviewing the contract

Issues found

Severity	Number os issues found
High	2
Medium	0
Low	0
Info	1
Total	3

Findings

High

[H-1] TITLE Password stored on chain makes it visible to everyone and no longer private asset

Description: Data stored on-chain is not private anymore but available for the public to read. The `PasswordStore::s_password` variable is supposed to be a secret and accessed through only the `PasswordStore::getPassword` function, which is intended to be only called by the owner of the contract

Impact: Everyone can read the password, and thats not safe for the protocol

Proof of Concept: 1. Create a locally running chain

```
make anvil
```

2. Deploy the contract to the chain

```
make deploy
```

3. run the storage tool

```
cast storage <address> --rpc-url <rpc-url>
```

you'll get an output that looks like this 0x6d7950617373776f726400000000000000000000000000000000000000000014

you can then parse that hex to a string with:

```
cast parse-bytes32-string 0
    x6d7950617373776f726400000000000000000000000000000000000000000014
```

and get an output of

```
myPassword
```

Recommended Mitigation: Due to this, the overall architecture of the contract should be modified, one could encrypt the password off-chain, and then store the encrypted password on-chain. This would require the user to remember another password off-chain to decrypt the password. However, you'd also likely want to remove the view function as you wouldn't want the user to accidentally send a txn with the password that decrypts your password.

[H-2] `PasswordStore` has no access controls so anyone couls change the password

Description: This external function is supposed to allow only owner to set an new Password

```
    function setPassword(string memory newPassword) external {
         s_password = newPassword;
         emit SetNetPassword();
     }
```

Impact: Anyone can change the Password of the contract

Proof of concept: Add the following to the PasswordStore.t.sol

```
function test_anyone_can_set_password(address randomAddress) public{
        vm.assume(randomAddress != owner);
        vm.prank(randomAddress);
        string memory expectedPassword = "myPassword";
        passwordStore.setPassword(expectedPassword);

        vm.prank(owner);
        string memory actualPassword = passwordStore.getPassword();
        assertEq(actualPassword, expectedPassword);
```

```
    }
```

Recommended Mitigation: Add an access control conditional to the `setPassword` function, you can make it a modifier too so you can attach it to most functions.

Informational

[I-1] There is no parameter as indicated in the natspec

Description:

```
function getPassword() external view returns (string memory) {
        if (msg.sender != s_owner) {
            revert PasswordStore__NotOwner();
        }
        return s_password;
    }
```

The `PasswordStore::getPassword` function does not indicate any param but the natspec says it should be `getPassword(string)`

Impact: natspec is misleading

Proof of Concept: check natspec notes

Recommended Mitigation: remove the incorrect natspec line

```
-     * @param newPassword The new password to set.
```